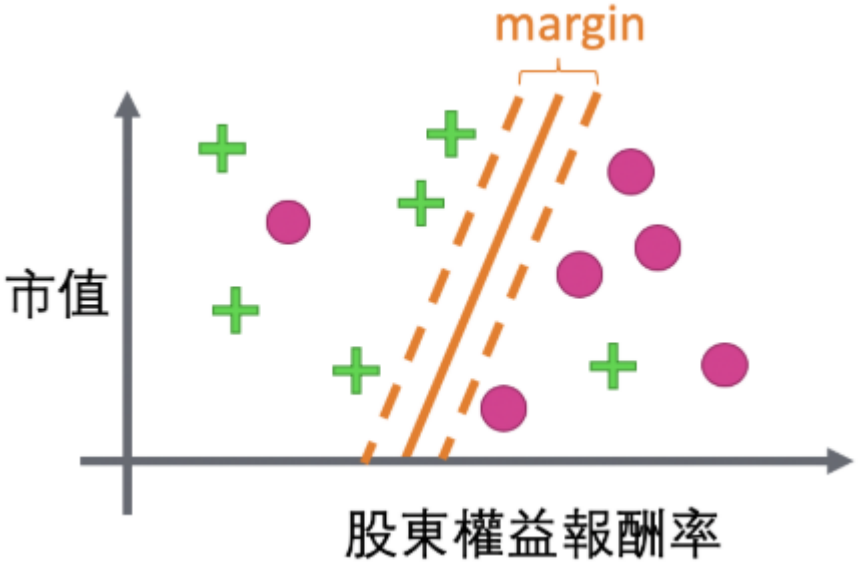
分類模型：支持向量機 (Support Vector Machine)

Some slides were borrowed from Andrew Moore's PowetPoint slides on SVMs. Andrew's PowerPoint repository is here: http://www.cs.cmu.edu/~awm/tutorials . Comments and corrections gratefully received.

範例



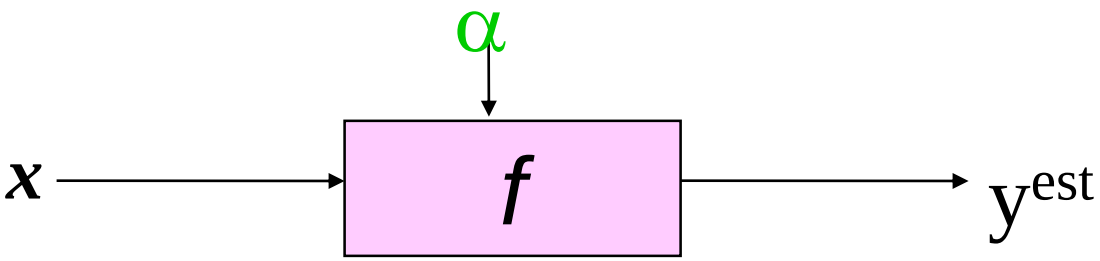
● 一個月後上漲的股票

＋ 一個月後下跌的股票

Support Vector Machine

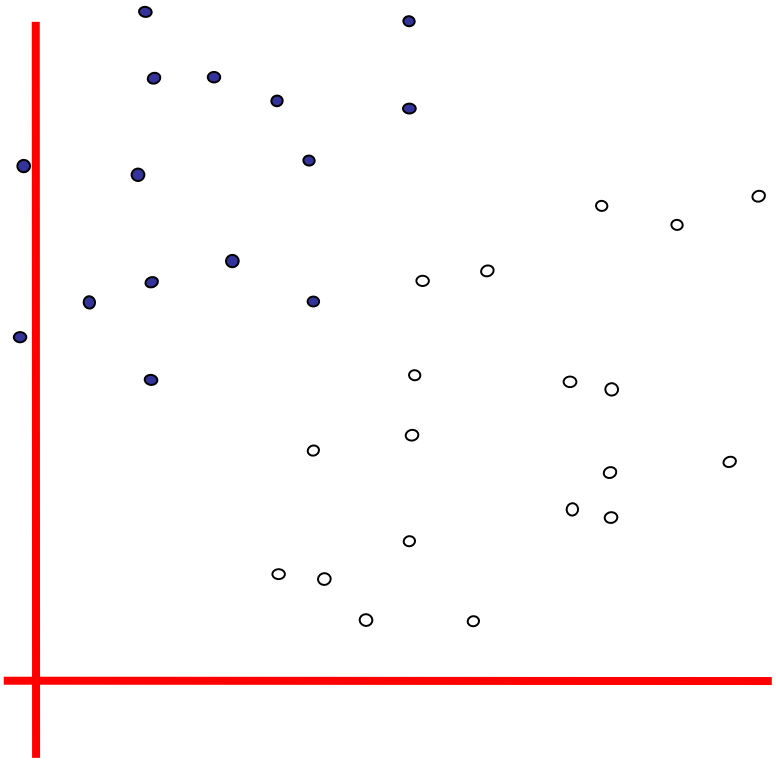
- Very popular ML technique
  - Became popular in the late 90s (Vapnik 1995; 1998)
  - Invented in the late 70s (Vapnik, 1979)
- Controls complexity and overfitting, so works well on a wide range of practical problems
- Can handle high dimensional vector spaces, which makes feature selection less critical
- Very fast and memory efficient implementations, e.g., svm_light
- Not always the best solution, especially for problems with small vector spaces

線性分類器 (Linear Classifiers)



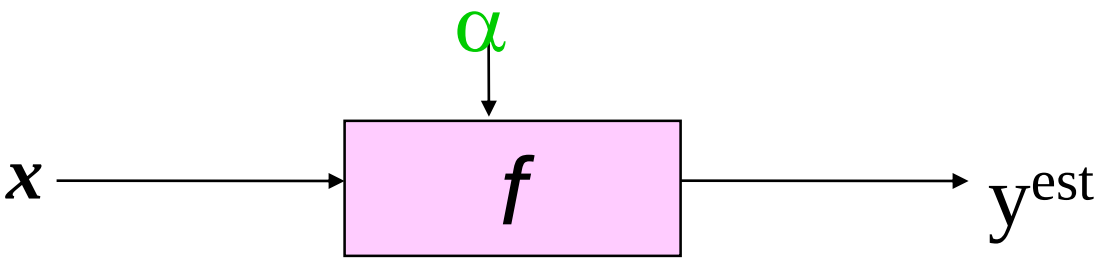
- denotes +1
- denotes -1

$f(x,w,b) = sign(w.\ x - b)$



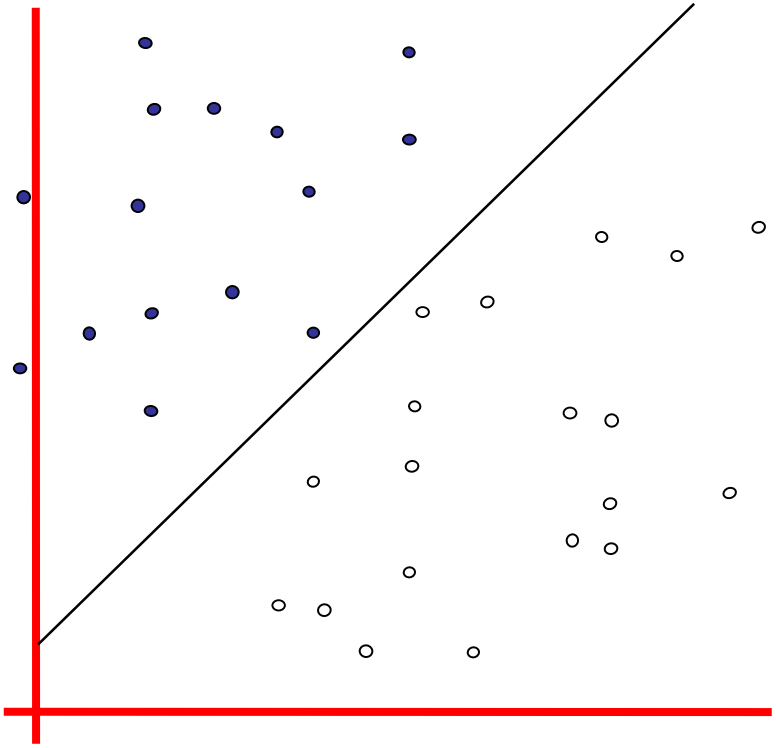
How would you classify this data?

線性分類器 (Linear Classifiers)



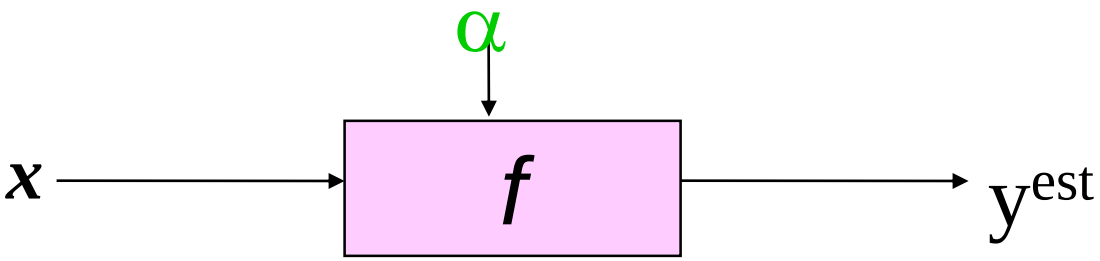
- denotes +1
- denotes -1

$f(x,w,b) = sign(w.\ x - b)$



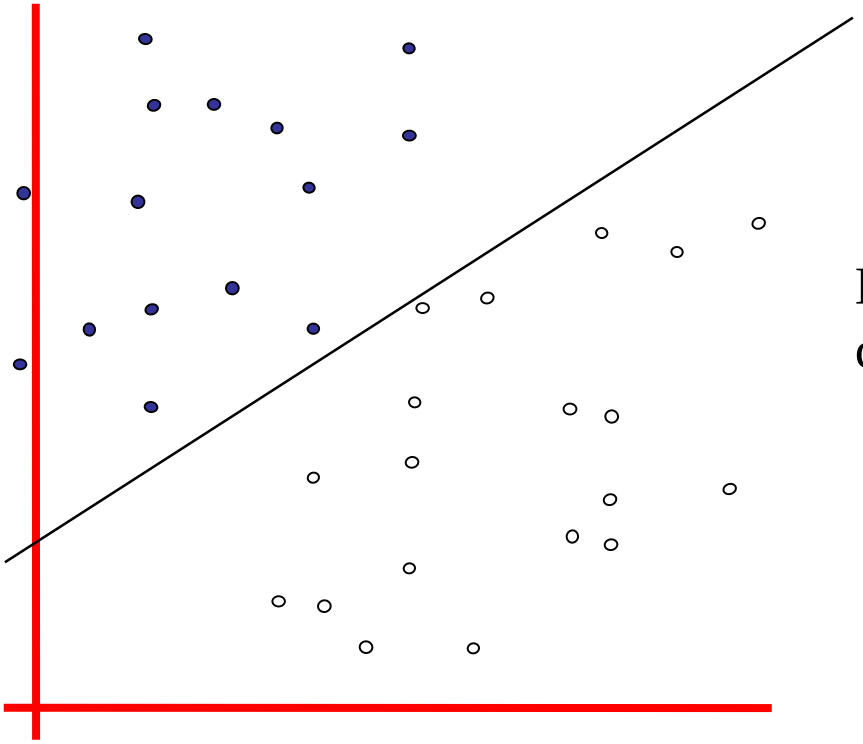
How would you classify this data?

線性分類器 (Linear Classifiers)



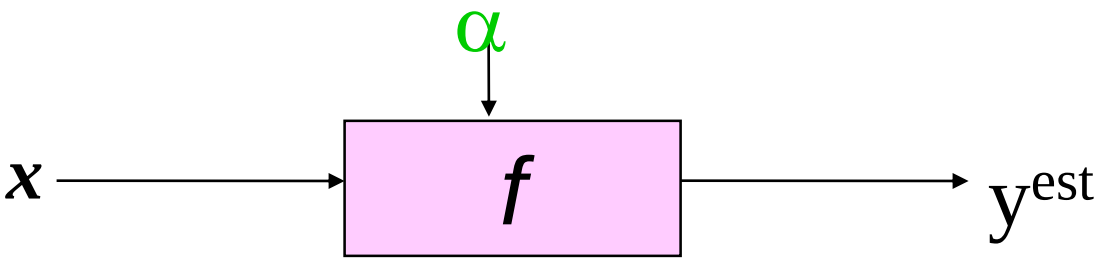
- denotes +1
- denotes -1

$f(x,w,b) = sign(w.\ x - b)$

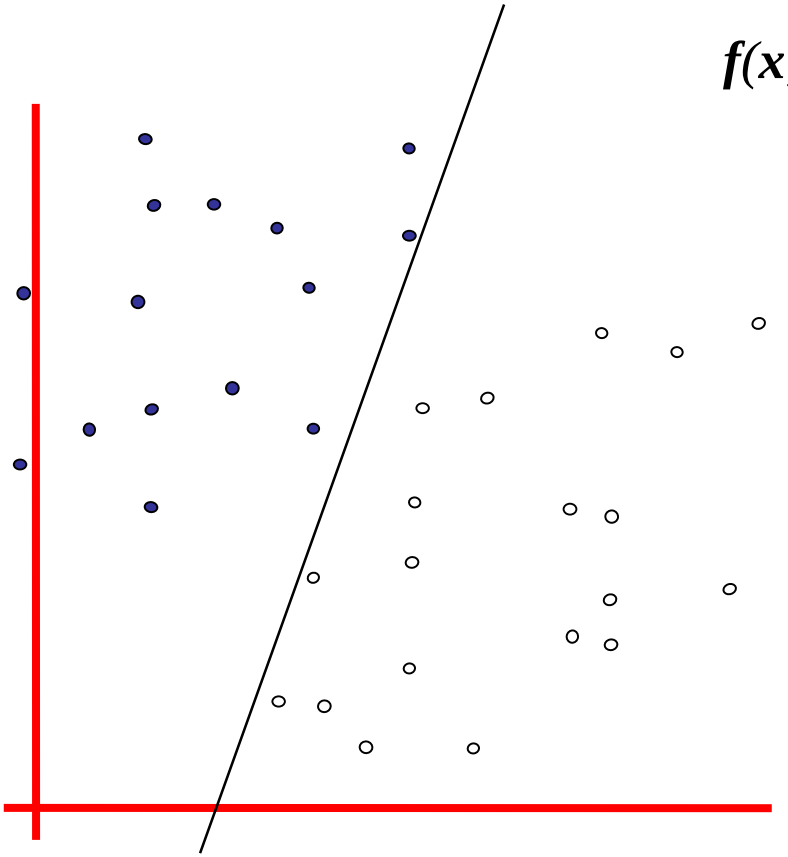


How would you classify this data?

線性分類器 (Linear Classifiers)



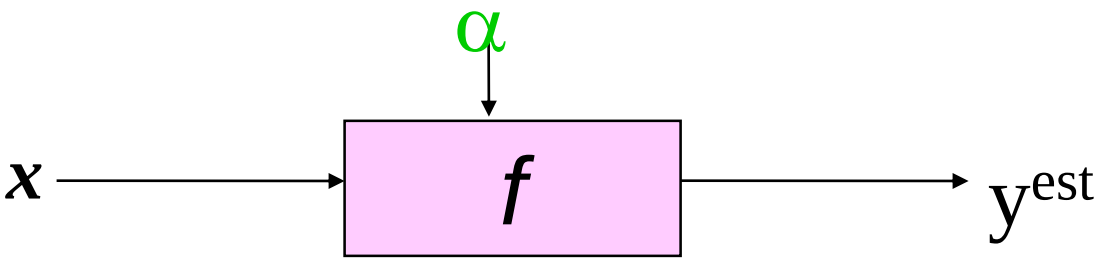
- denotes +1
- denotes -1



$f(x,w,b) = sign(w.\ x - b)$

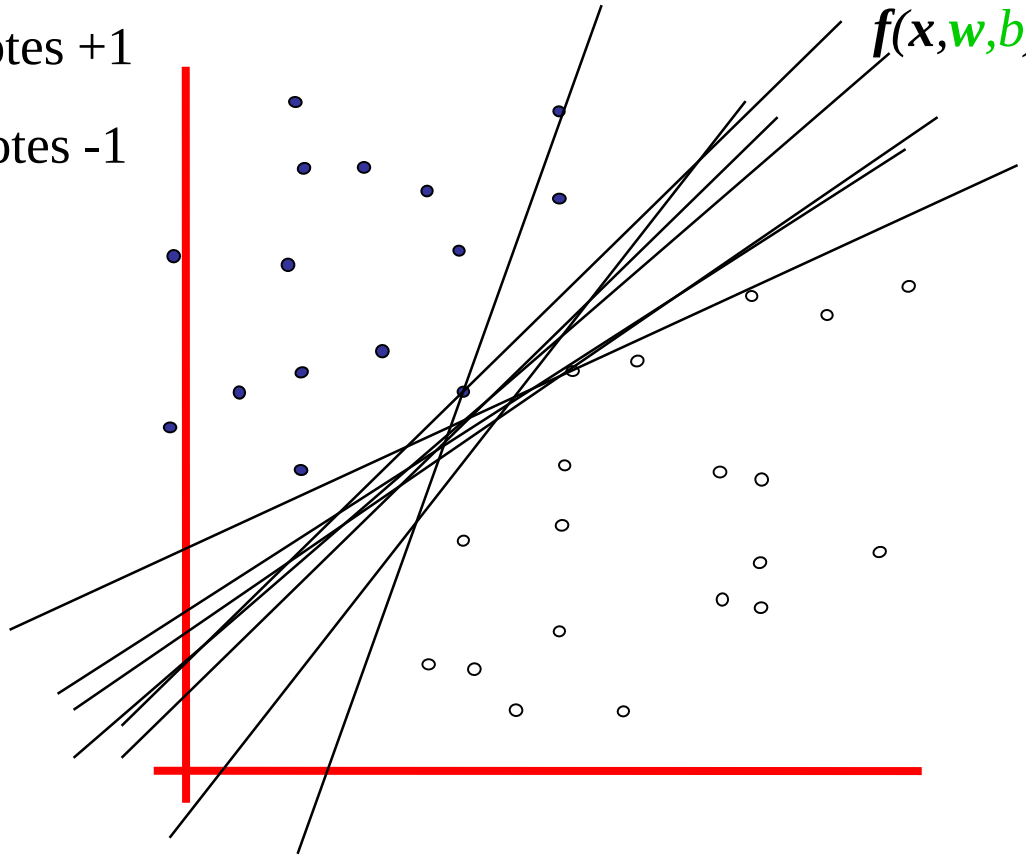
How would you classify this data?

線性分類器 (Linear Classifiers)



- denotes +1
- denotes -1

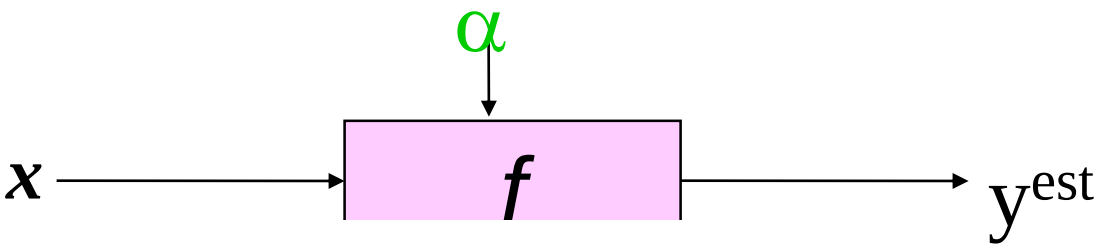
$f(x,w,b) = sign(w.\ x - b)$



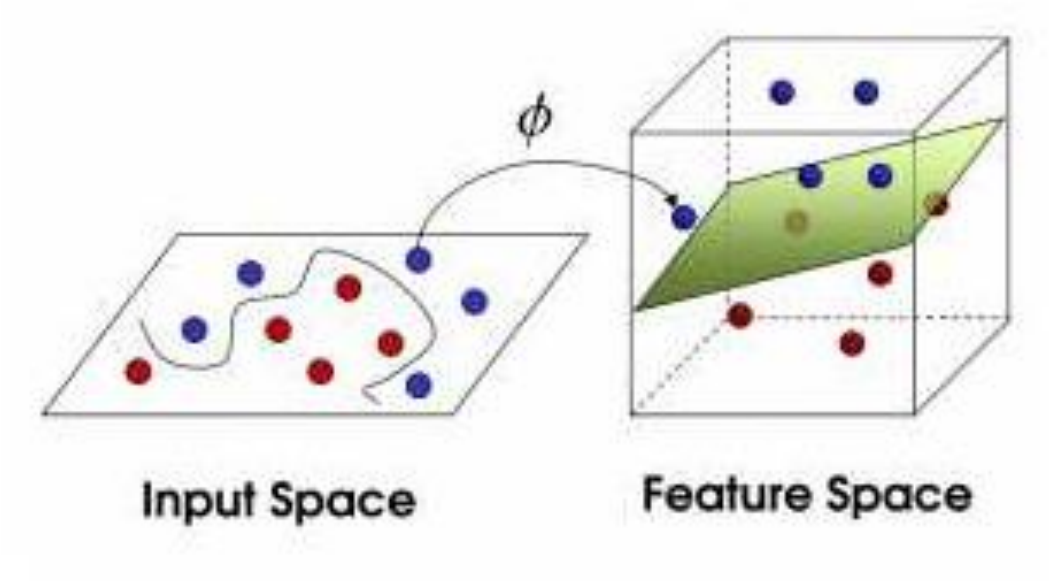
Any of these would be fine..

..but which is best?

如何分類？

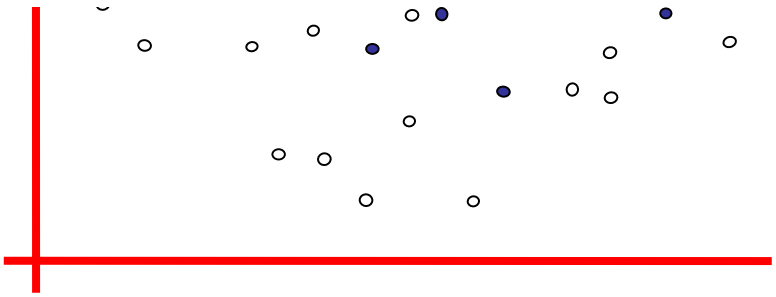


- den
  - der

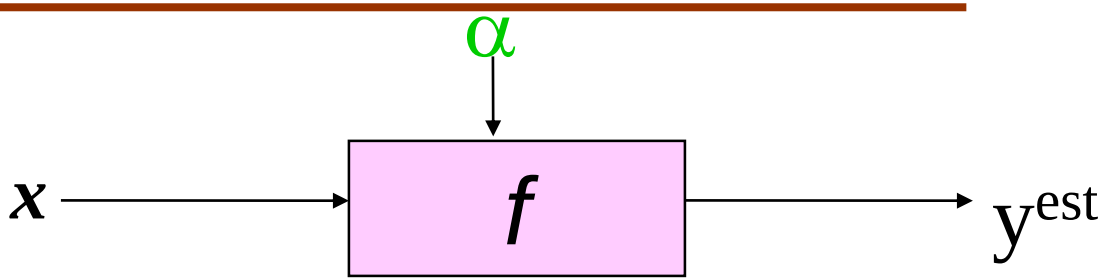


w. x - b)

lo you do
ssify?

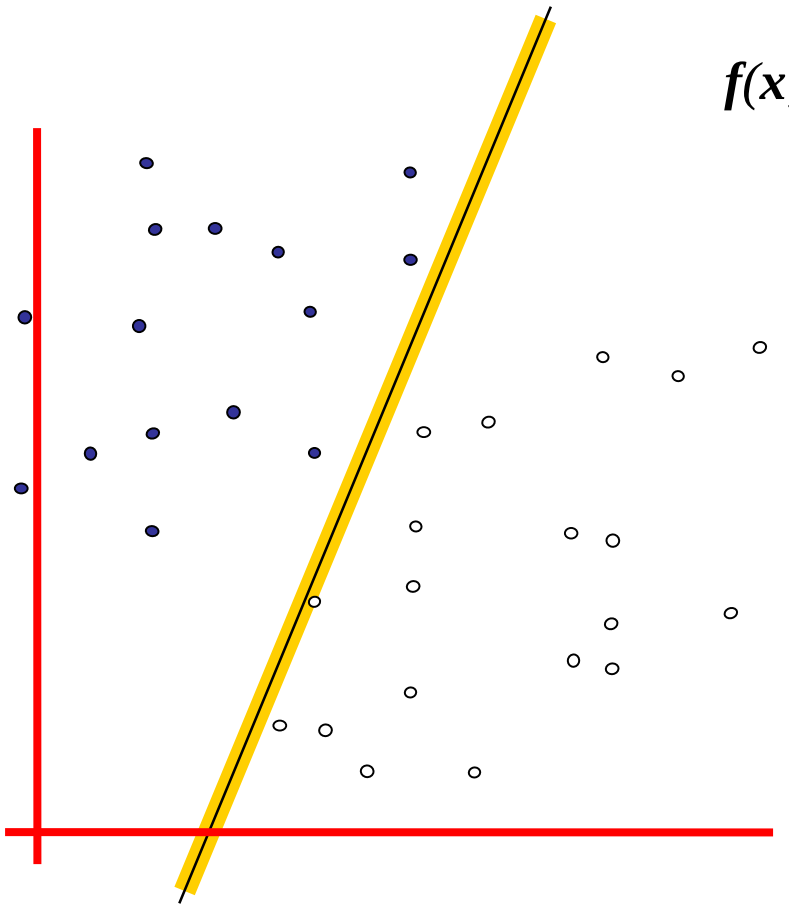


分類的Margin



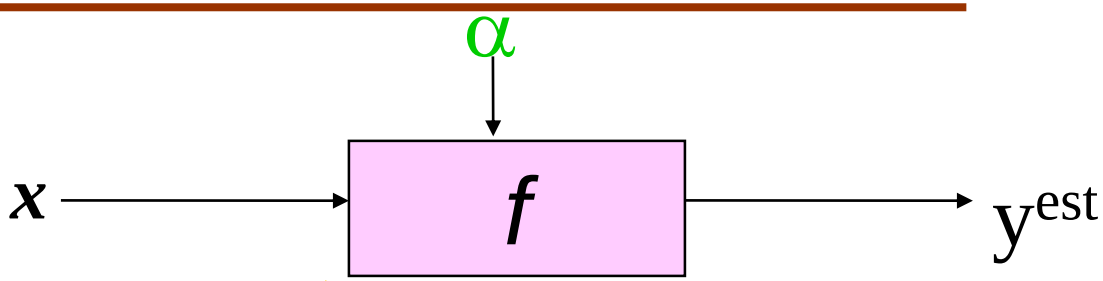
$f(x,w,b) = sign(w.\ x - b)$

- denotes +1
- denotes -1



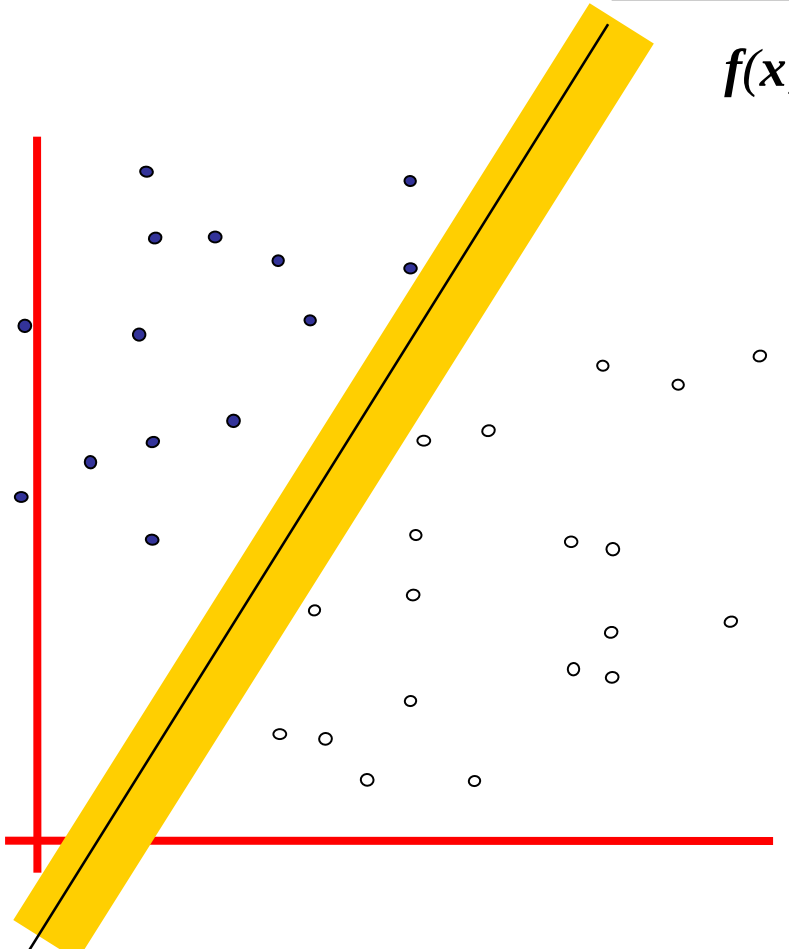
訂義線性分類器的margin，在沒有碰到資料點之內增加寬度。

讓Margin最大化



$f(x,w,b) = sign(w.\ x - b)$

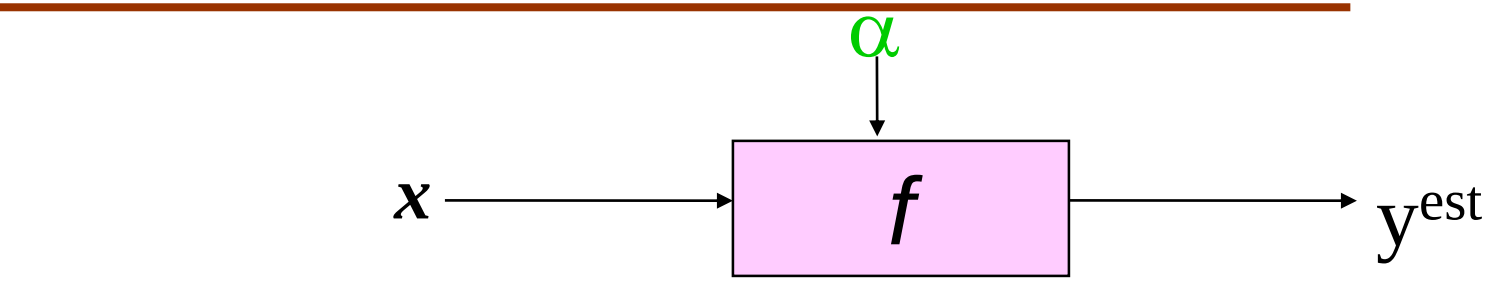
- denotes +1
- denotes -1



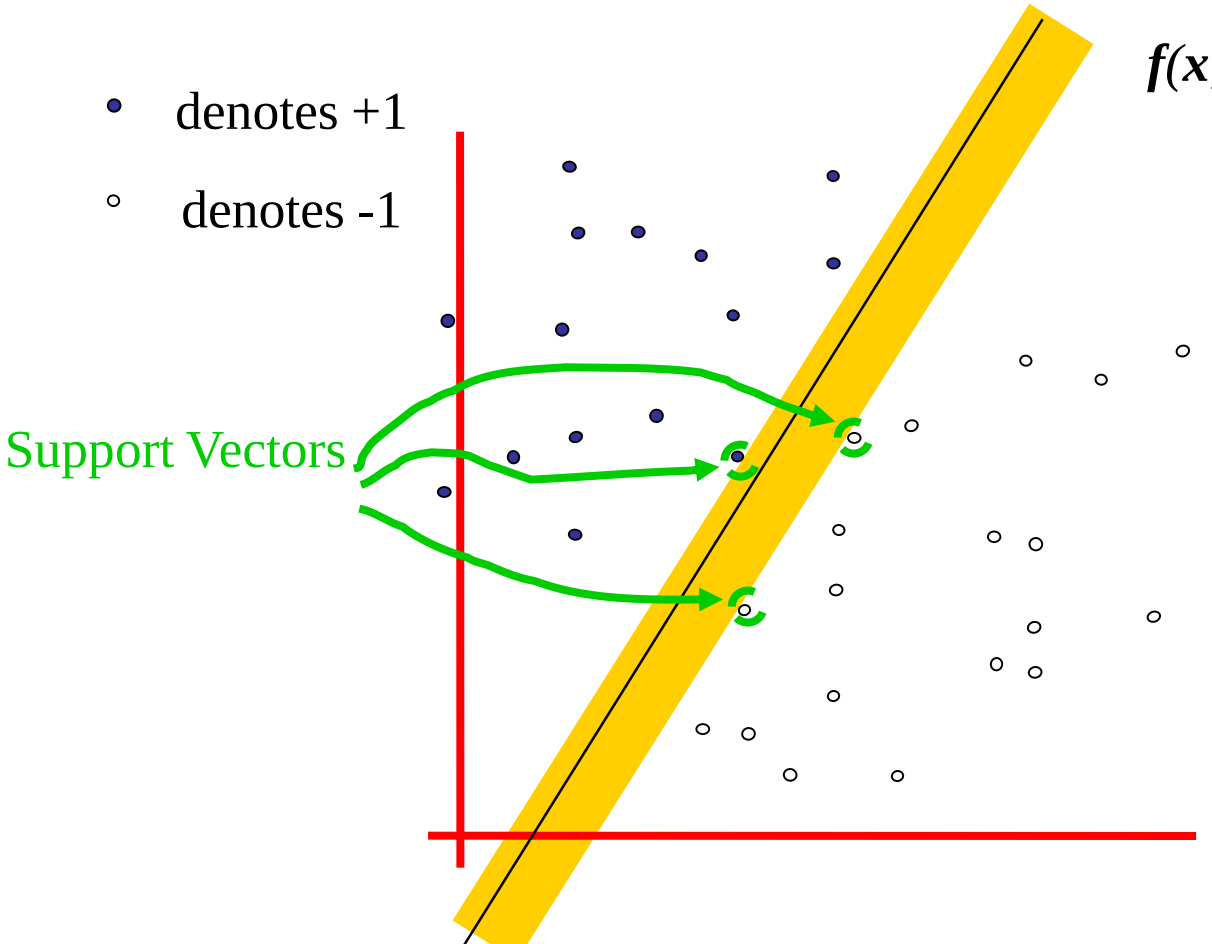
增加寬度到碰到最近的點，目標是要最大化這個寬度

Linear SVM

讓Margin最大化



$f(x,w,b) = sign(w.\ x - b)$

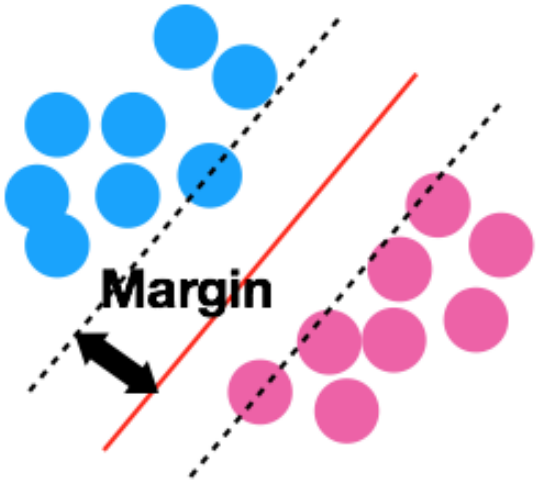


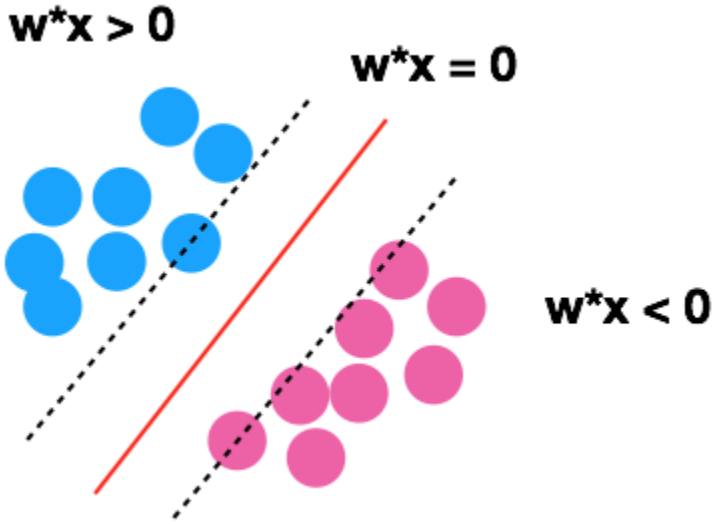
碰到Margin的點即為Support Vector (支援向量)，利用支援向量還算出最大的Margin

Linear SVM

讓Margin最大化的數學表示

- 假設紅線是w*x = 0 在紅線上方的區域就是w*x >0 紅線下方的區域就是w*x <0。

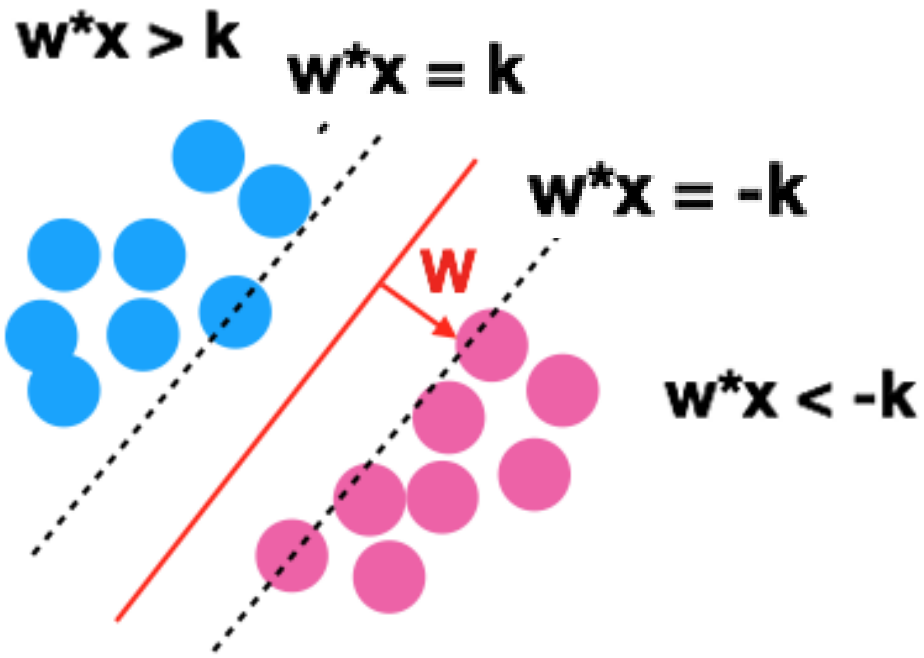




https://medium.com/jameslearningnote/%E8%B3%87%E6%96%99%E5%88%86%E6%9E%90-%E6%A9%9F%E5%99%A8%E5%AD%B8%E7%BF%92-%E7%AC%AC3-4%E8%AC%9B-%E6%94%AF%E6%8F%B4%E5%90%91%E9%87%8F%E6%A9%9F-support-vector-machine-%E4%BB%8B%E7%B4%B9-9c6c6925856b

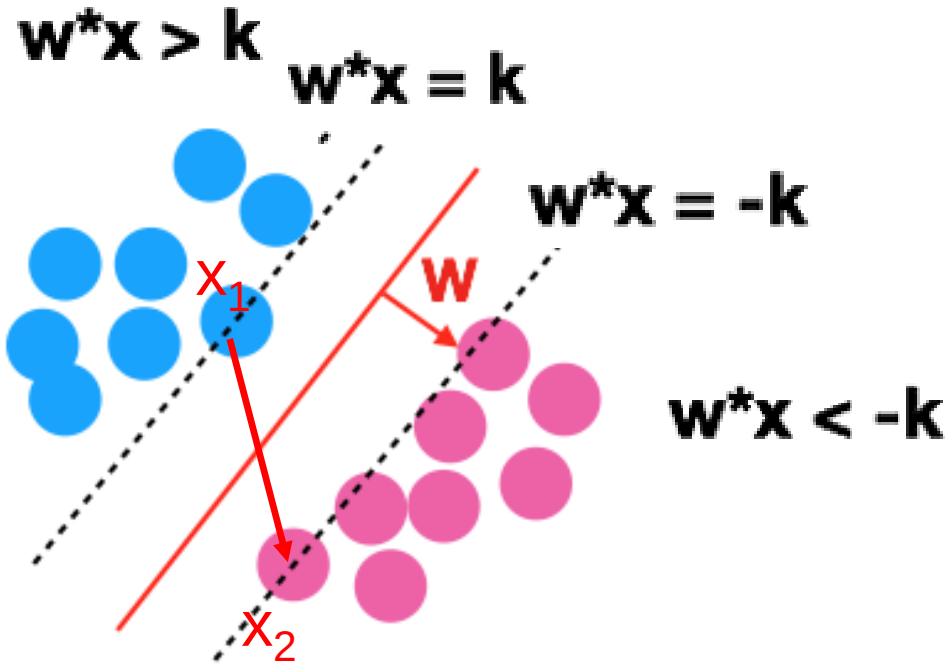
讓Margin最大化的數學表示

- 同理類推來看在左邊虛線上方的區域是w*x>k
- 在右邊虛線下方的區域是w*x< k
- 虛線中間不會有資料點。



讓Margin最大化的數學表示

- 虛線上的點X1,X2 其實就是所謂的支援向量(Support vector)
- 利用支援向量來算出Margin，並最大化Margin
- 高中數學的知識將X1向量-X2向量得到的向量投影到W就可以了
- 在Y*(W*X) ≥k 的條件下(虛線中間沒有點)，來最大化margin



$$\text{Margin} = \frac{W * \overrightarrow{X1 - X2}}{2\|w\|} = \frac{2k}{2\|w\|}$$

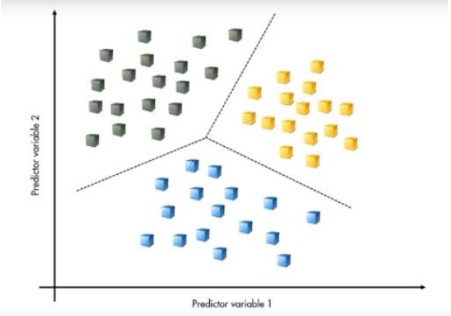
$$\text{Condition} = y^*(w^*x) >= k$$

SVM 的效能

- 可以處理很多維度的資料 (e.g., 100K features)
- 學習速度快Relatively fast
- 分類效果很好
  - 在手寫文字識別上是目前最好的分類器

二元分類 vs. 多元分類

- SVMs 只能做二元分類
- 兩種方法可以做多元分類:
  - 一對多 (One-vs-all):
    - 魚 vs 不是魚
    - 哺乳類 vs 不是哺乳類
    - 選擇可以得到較好分類結果的分法
  - 一對一 (One-vs-one): 任取兩類 classifiers that vote on results
    - 哺乳類 vs 魚
    - 哺乳類 vs 爬行動物



Lab: SVM

安裝環境

- pip install scikit-learn
- math 模組是標準庫中的，所以不用安裝

Environment Setup

- 開啟 Jupyter Notebook

Jupyter Notebook (Anaconda3)

應用程式

- 進入您存放檔案(資料集、code)的工作目錄
- 開啟svm.ipynb檔進入Python編譯環境

了解問題與資料型態
(Data Understanding)

Dataset

- **以熱壓爐溫度預測成化曲線的分類**
  - 2019全國智慧製造大數據分析競賽
  - 數據為熱壓爐成化加工過程所量測的溫度數據
    - 成化：複合材料加工至硬化的溫度
    - 依照機台型號可以分為 8 類
    - 目標：訓練multi-class分類模型，可以準確分 8 類

原始資料

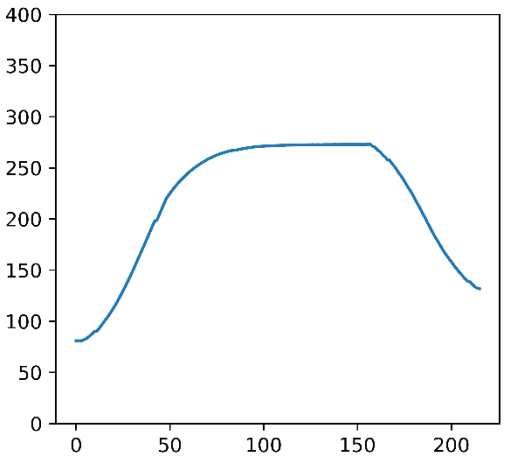
- 機台型號分為八組
  - G11,G15,G17,G19,G32,G34,G48,G49
  - 下圖為G11機台

G11-PTC13	G11-PTC14	G11-PTC15	G11-PTC16	G11-PTC17
Deg.F	Deg.F	Deg.F	Deg.F	Deg.F
65.9	65.8	64.2	64.9	66.0
65.9	65.8	64.2	64.9	66.0
65.9	65.8	64.2	64.9	66.0
65.9	65.8	64.2	64.9	66.0
66.6	67.2	65.7	65.6	67.3
68.0	68.9	67.5	66.4	68.6
69.4	70.8	69.6	67.1	70.0
71.7	73.6	72.3	68.3	72.0
74.3	76.9	75.1	69.6	74.0
77.1	80.0	78.4	70.9	76.4
79.9	83.0	81.3	72.4	78.5
80.1	83.2	81.6	72.6	78.8
83.5	87.0	85.3	74.7	81.4
86.6	90.5	89.2	76.8	84.3
90.2	94.7	93.3	79.1	87.1
93.4	98.0	97.5	81.3	90.6
97.3	102.1	101.5	84.1	94.0
100.9	106.6	106.0	86.9	97.5
104.7	110.2	110.5	89.7	101.0
108.2	114.3	114.9	93.0	104.8
112.1	118.7	119.5	96.3	108.8
116.2	123.2	124.3	99.6	112.7
120.3	127.9	128.8	103.1	116.8

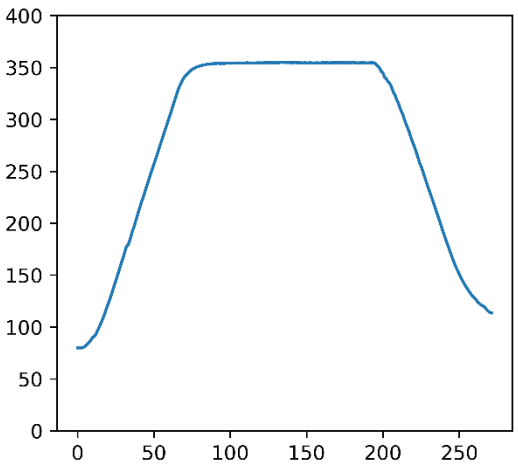
觀察資料

- 資料視覺化(Data visualization)
  - 由於是溫度資料，可以從曲線來觀察其特徵
  - 以三種機台資料為例，每種機台的溫度曲線都不盡相同
  - 有什麼想法嗎？要怎麼萃取特徵呢？

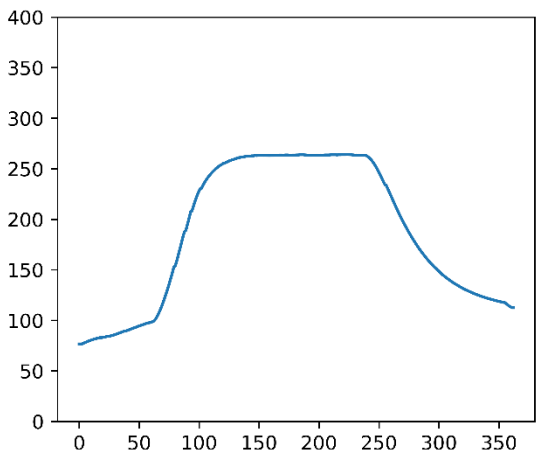
G11



G15



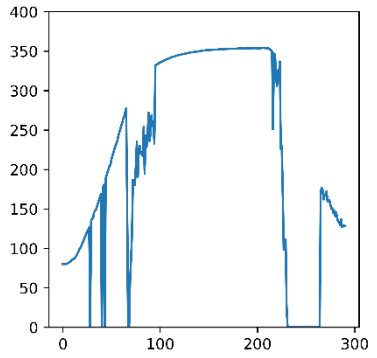
G32



前處理方法

Preprocessing

- 觀察是否有outlier
  - 如右圖，由於儀器沒量測到(出現0)導致噪音(noise)
  - Solution：斜率補值、中值補值等



- 檢查資料型態與長度
  - 發現每筆資料記錄溫度時間不等，將所有值補到一樣長，避免出現錯誤
  - 補完值，並將資料合併如附檔 - train.csv
  - 檔案的每個row為一機台之成化曲線，label也轉成0-7方便下一步程式撰寫

Import

1. import所有要用到的package

```
#import package
import pandas as pd
import numpy as np
from sklearn.model_selection import KFold
from sklearn import ensemble, metrics
from sklearn.metrics import accuracy_score
from sklearn.utils import shuffle
```

Data Labeling

2. 用pandas讀取csv檔

```
# 開啟 CSV 檔案
dataset= pd.read_csv('train.csv')
testset= pd.read_csv('test.csv')
```

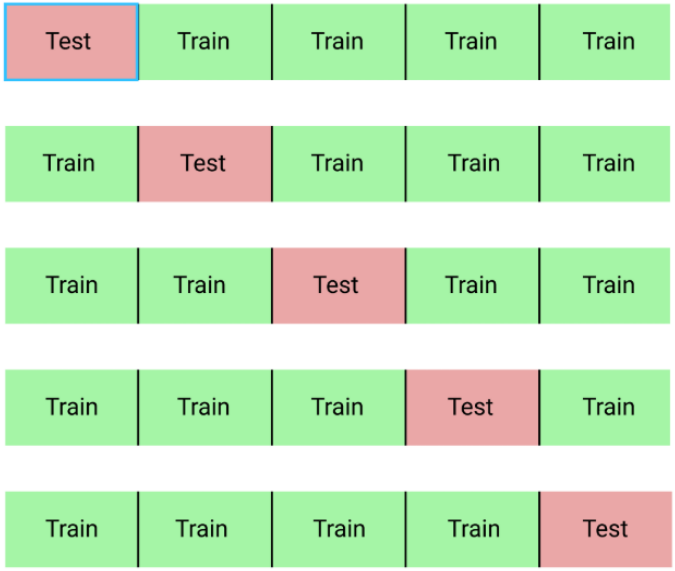
3. 將資料打散(shuffle)避免overfitting
   將label另外存，方便後續操作

```
#將dataset shuffle
dataset = shuffle(dataset)
label = dataset.label
dataset = dataset[dataset.columns[:449]]
```

模型交叉驗證
(Cross-validation)

交叉驗證

- 加入交叉驗證，保證模型的可靠度



交叉驗證示意圖

10-fold cross-validation

```
#10-fold cross-validation
kfold = KFold(10, True)
predicted = []
expected = []
```

拆成train,test

- Kfold.split會將dataset拆成train,test

```
[8]: for train, test in kfold.split(dataset):
         X_train= dataset.iloc[train]
         Y_train = label.iloc[train]
```

```
[9]: print("TRAIN:", train, "TEST:", test)
```

```
TRAIN: [   0    2    3 ... 1738 1739 1740] TEST: [   1    4    9   22   23   29   43   61   97  126  139  158  167  168
  178  185  189  191  202  209  223  225  228  248  259  261  280  284
  294  298  346  354  365  367  384  410  412  418  440  448  459  481
  514  515  517  518  519  530  542  543  547  555  572  592  600  612
  617  618  630  633  654  657  679  689  690  716  725  754  762  767
  775  786  794  804  805  809  814  825  828  842  845  848  878  882
  889  906  907  914  935  970  972  973  974 1003 1026 1029 1032 1045
 1053 1061 1066 1082 1084 1091 1093 1095 1100 1107 1121 1126 1150 1151
 1153 1166 1176 1183 1201 1220 1222 1235 1238 1242 1254 1255 1258 1268
 1275 1277 1279 1285 1311 1319 1320 1329 1334 1337 1343 1345 1366 1376
 1379 1384 1409 1413 1417 1422 1427 1436 1438 1439 1449 1450 1463 1474
 1477 1489 1497 1498 1507 1511 1534 1543 1544 1589 1597 1600 1608 1612
 1625 1653 1664 1682 1689 1731]
```

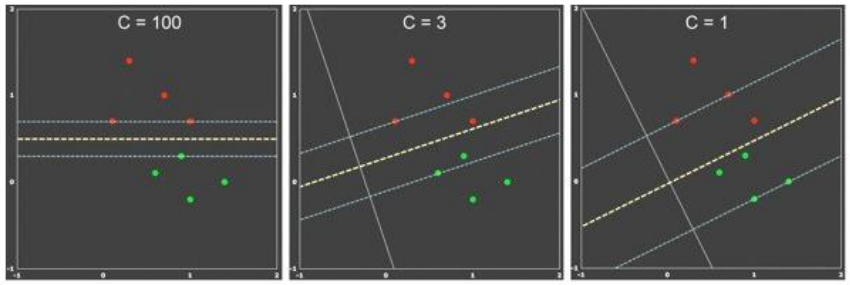
sklearn.svm.SVC

```
class sklearn.svm.SVC(*, C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=0.001,
cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', break_ties=False,
random_state=None)                                                                                                    [source]
```

- SVM:
  - Margin越大越好
  - Misclassification rate越小越好

⇒ Trade off，要選擇哪一個?

在訓練時，misclassification rate越小，
不代表測試時，也會得到小的misclassification rate
=>SVM選擇margin越大越好



Change in margin with change in C

C: 用來控制margin的參數
=>C越大，margin越小

選擇分類模型

- 以SVM為例，由於資料不只兩類，需要用multi-class SVM
  - 可以選擇one-against-one / one-against-rest

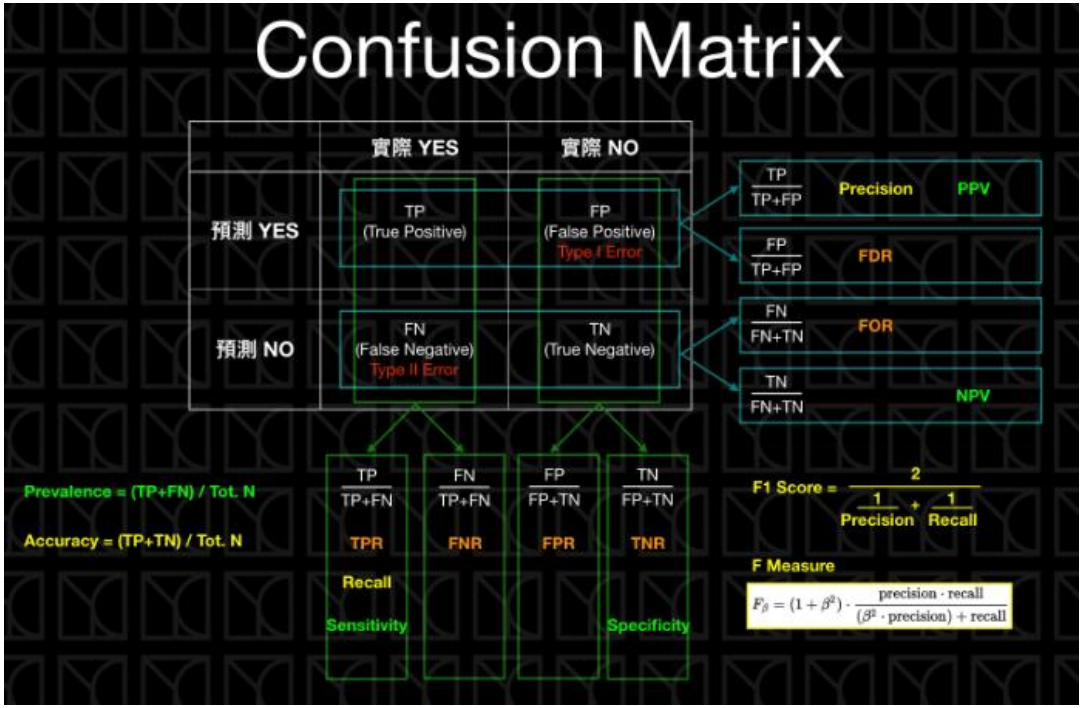
SVM分類器

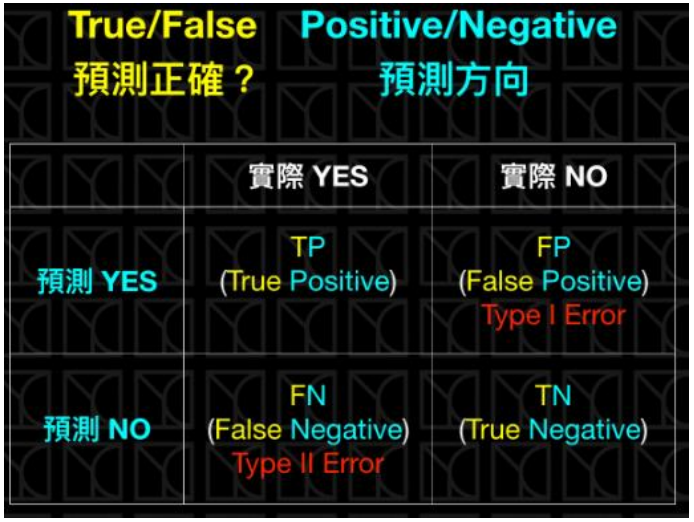
```
#svm
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import SVC

#10-fold cross-validation
kfold = KFold(10, True)
predicted = []
expected = []

#會出現warning為正常現象，不用理會，也可以另外寫code去ignore.
for train, test in kfold.split(dataset):
    X_train= dataset.iloc[train]
    Y_train = label.iloc[train]
    X_test = dataset.iloc[test]
    Y_test = label.iloc[test]
    svm = OneVsRestClassifier(SVC(gamma='scale')).fit(X_train,Y_train)
    expected.extend(Y_test)
    predicted.extend(svm.predict(X_test))
```

將資料拆成train和test

選擇one-against-rest的SVM分類器

Confusion Matrix





- 前面的True和False代表預測本身的結果是正確還是不正確的
- 而後面的Positive和Negative則是代表預測的方向是正向還是負向的。

https://www.ycc.idv.tw/confusion-matrix.html

範例

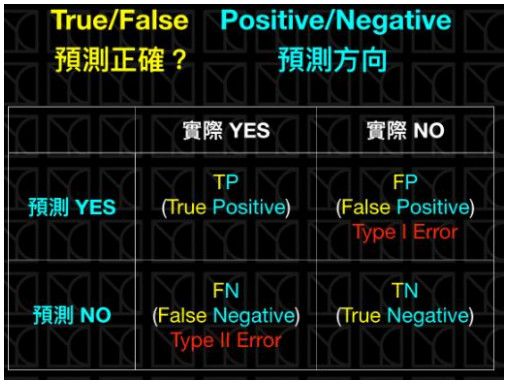
- 你的手機iphne指紋解鎖

	實際 Yes	實際 No
預測 Yes	a	d
預測 No	c	b

(a) iphone正確解鎖你的指紋
(b) iphone無法解鎖你朋友的指紋
(c) iphone無法解鎖你的指紋
(d) iphone解鎖你朋友的指紋

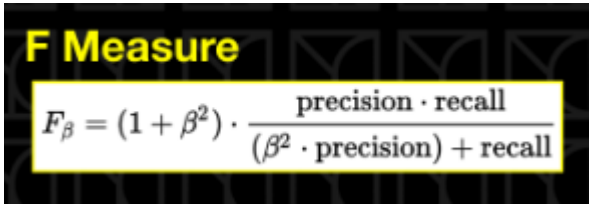
Confusion Matrix

- Accuracy= (TP+TN)/(TP+FP+FN+TN)
  - 所有情形下判斷正確有多少
  - 在實際正向的情況很少時會失效，如信用卡盜刷，模型可輕易達99%以上
- Recall(召回率) = TP/(TP+FN)
  - 在實際情形為正向的狀況下，預測「能召回多少」實際正向的答案
  - 廣告投放
- Precision(準確率) = TP/(TP+FP)
  - 在預測正向的情形下，實際的「精準度」是多少
  - 門禁系統



Confusion Matrix

- F Measure



- β=1時的特例 => F1-score
  - 代表Precision和Recall都同等重要，採用F1-Score
  - F1-score = 2 * Precision * Recall / (Precision + Recall)
- 若是precision比較重要，β小一點
- 若是recall比較重要，β大一點
- β=0時，即為precision
- β無限大時，即為recall

Average

- average=micro
  - Compute f1 by considering total true positives, false negatives and false positives (no matter of the prediction for each label in the dataset)
- average=macro
  - Compute f1 for each label, and returns the average without considering the proportion for each label in the dataset.
- average=weighted
  - Compute f1 for each label, and returns the average considering the proportion for each label in the dataset.
- average=samples
  - Compute f1 for each instance, and returns the average.

計算模型準確度

- 計算模型準確度、confusion matrix

```
print("Macro-average: {0}".format(metrics.f1_score(expected,predicted,average='macro')))
print("Micro-average: {0}".format(metrics.f1_score(expected,predicted,average='micro')))
print(metrics.classification_report(expected,predicted))
print(metrics.confusion_matrix(expected, predicted))
accuracy = accuracy_score(expected, predicted)
print("Accuracy: %.2f%%" % (accuracy * 100.0))

print("Average = macro")
print('precision:',metrics.precision_score(expected, predicted,average='macro'))
print('recall:',metrics.recall_score(expected, predicted,average='macro'))
print('F1-score:',metrics.f1_score(expected, predicted,labels=[1,2,3,4],average='macro'))

print("\n")
print("Average = micro")
print('precision:', metrics.precision_score(expected, predicted, average='micro'))
print('recall:',metrics.recall_score(expected, predicted,average='micro'))
print('F1-score:',metrics.f1_score(expected, predicted,labels=[1,2,3,4],average='micro'))

print("\n")
print("Average = weighted")
print('precision:', metrics.precision_score(expected, predicted, average='weighted'))
print('recall:',metrics.recall_score(expected, predicted,average='micro'))
print('F1-score:',metrics.f1_score(expected,predicted,labels=[1,2,3,4],average='weighted'))
```

SVM Result

- Result from SVM
  - Accuracy: 99.02%

```
Macro-average: 0.990224871165917
Micro-average: 0.9902354968408961
              precision    recall  f1-score   support

           0       1.00      0.97      0.99       145
           1       1.00      0.99      1.00       207
           2       1.00      0.97      0.99       119
           3       1.00      1.00      1.00       238
           4       0.99      0.99      0.99       256
           5       1.00      1.00      1.00       264
           6       1.00      0.99      0.99       240
           7       0.95      0.99      0.97       272

   micro avg       0.99      0.99      0.99      1741
   macro avg       0.99      0.99      0.99      1741
weighted avg       0.99      0.99      0.99      1741

[[141   0   0   0   0   0   0   4]
 [  0 205   0   0   0   0   0   2]
 [  0   0 116   0   0   0   0   3]
 [  0   0   0 238   0   0   0   0]
 [  0   0   0   0 253   0   0   3]
 [  0   0   0   0   0 264   0   0]
 [  0   0   0   0   0   0 237   3]
 [  0   0   0   0   2   0   0 270]]
Accuracy: 99.02%
Average = macro
precision: 0.9924406604747162
recall: 0.9882462727680715
recall: 0.9882462727680715
F1-score: 0.9931487344522549


Average = micro
precision: 0.9902354968408961
recall: 0.9902354968408961
F1-score: 0.9938800489596082


Average = weighted
precision: 0.9906239904589667
recall: 0.9902354968408961
F1-score: 0.9938672003987704
```

下一步

以上提到的方法，你可以繼續嘗試提高準確度

1. 加強前處理的方法
   - 除了將資料填補至一樣長度，還可以嘗試其他方法
   - 是否將所有資料放入，也可以排除outlier
2. 使用其他模型
   - scikit-learn內建非常多的模型，而且只要1,2行就可以引入
   - 參考: https://scikit-learn.org/stable/supervised_learning.html